Graph-based Modelling of Students' Interaction Data from Exploratory Learning Environments

Alexandra Poulovassilis
London Knowledge Lab
Birkbeck, Univ. of London
ap@dcs.bbk.ac.uk

Sergio Gutierrez-Santos
London Knowledge Lab
Birkbeck, Univ. of London
sergut@dcs.bbk.ac.uk

Manolis Mavrikis
London Knowledge Lab
UCL Institute of Education
m.mavrikis@lkl.ac.uk

ABSTRACT

Students' interaction data from learning environments has an inherent temporal dimension, with successive events being related through the "next event" relationship. Exploratory learning environments (ELEs), in particular, can generate very large volumes of such data, making their interpretation a challenging task. Using two mathematical microworlds as exemplars, we illustrate how modelling students' event-based interaction data as a graph can open up new querying and analysis opportunities. We demonstrate the possibilities that graph-based modelling can provide for querying and analysing the data, enabling investigation of student-system interactions and leading to the improvement of future versions of the ELEs under investigation.

Keywords

Exploratory Learning Environments, Interaction Data, Graph Modelling

1. INTRODUCTION

Much recent research has focussed on *Exploratory Learning Environments* (ELEs) which encourage students' open-ended interaction within a knowledge domain, coupled with intelligent techniques that aim to provide pedagogical support to ensure students' productive interaction [9]. The data gathered from students' interactions with such ELEs provides a rich source of information for both pedagogical and technical research, to help understand how students are using the ELE and how the intelligent support that it provides may be enhanced to better support students' learning.

In this paper, we consider how modelling students' event-based interaction data as a *graph* makes possible graph-based queries and analyses that can provide insights into the ways that students are using the affordances of the system and the effects of system interventions on students' behaviour. Our case studies are two intelligent ELEs: the MiGen system, that aims to foster 11-14 year old students' learning of algebraic generalisation [15]; and the iTalk2Learn system that aims to support 8-10 year old students' learning of fractions [7]. Both systems provide students with mathematical microworlds in which they undertake construction tasks: in MiGen creating 2-dimensional tiled models using a tool called *eXpresser* and in iTalk2learn creating fractions using the *FractionsLab* tool. In eXpresser, tasks typically require the construction of several models, moving from specific models involving specific numeric values to a general model involving the use of one or more variables; in parallel, students are asked to formulate algebraic rules specifying the number of tiles of each colour that are needed to fully colour their models. In FractionsLab, tasks require the construction, comparison and manipulation of fractions, and students are encouraged to talk aloud about aspects of their constructions, such as whether two fractions are equivalent.

Both systems include intelligent components that provide different levels of feedback to students, ranging from unsolicited prompts and nudges, to low-interruption feedback that students can choose to view if they wish. The aim of this feedback is to balance students' freedom to explore while at the same time providing sufficient support to ensure that learning is being achieved [9]. The intelligent support is designed through detailed cognitive task analysis and Wizard-of-Oz studies [13], and it relies on meaningful *indicators* being detected as students are undertaking construction tasks. Examples of such indicators in MiGen are 'student has made a building block' (part of a model), 'student has unlocked a number' (i.e. has created a variable), 'student has unlocked too many numbers for this task'; while examples of such indicators in FractionsLab are 'student has created a fraction', 'student has changed a fraction' (numerator or denominator), 'student has released a fraction' (i.e. has finished changing it).

Teacher Assistance tools can subscribe to receive real-time information relating to occurrences of indicators for each student, and can present aspects of this information visually to the teacher [8]. Indicators are either *task independent* (TI) or *task dependent* (TD). The former refer to aspects of the student's interaction that are related to the microworld itself and do not depend on the specific task the student is working on, while the latter require knowledge of the task the student is working on, may relate to combinations of student actions, and their detection requires intelligent reasoning to be applied (a mixture of case-based, rule-based and probablistic techniques). Detailed discussions of MiGen's TI

and TD indicators and how the latter are inferred may be found in [8].

In this paper we explore how graph-based representation of event-based interaction data arising from ELEs such as MiGen and FractionsLab can aid in the querying and analysis of such data, with the aim of exploring both the behaviours of the students in undertaking the exploratory learning tasks set and the effectiveness of the intelligent support being provided by the system to the students. Data relating to learning environments has often been modelled as a graph in previous work, for example in [10] for providing support to moderators in e-discussion environments; in [16, 18] for supporting learning of argumentation; in [17] for modelling data and metadata relating to episodes of work and learning in a lifelong learning setting; in [1] for learning path discovery as students "navigate" through learning objects; in [3] for recognising students' activity planning in ELEs; and in [23] for gaining better understanding of learners' interactions and ties in professional networks.

Previous work that is close to ours is the work on interaction networks and hint generation [6, 21, 20, 4, 5], in which the graphs used consist of nodes representing states within a problem-solving space and edges representing students' actions in transitioning between states. This approach targets learning environments where students are required to select and apply rules, and the interaction network aims to represent concisely information relating to students' problem-solving sequences in moving from state to state. Our focus here differs from this in that we are using graphs to model fine-grained event-based interaction data arising from ELEs. In our graphs, nodes are used to represent indicator occurrences (i.e. events, not problem states) and edges between such nodes represent the "next event" relationship. Also, rather than using the information derived from querying and analysing this data to automatically generate hints, our focus is on investigating how students are using the system and the effects of the system's interventions in order to understand how students interact with the ELEs and improve their future versions.

2. GRAPH-BASED MODELLING

Figure 1 illustrates our Graph Data Model for ELE interaction data. We see two classes of nodes: Event — representing indicator occurrences; and EventType — representing different indicator types. The instances of the Event class are occurrences of indicators that are detected or generated by the system as each student undertakes a task. We see that instances of Event have several attributes: dateTime: the date and time of the indicator occurrence; userID: the student it relates to; sessionID: the class session that the student was participating in at the time; taskID: the taskID that the student was working on; and constrID: the construction that the student was working on[1].

[1]The model in Fig. 1 focusses on the interaction data. The full data relating to ELEs such as eXpresser and FractionsLab would also include classes relating to users, tasks, sessions and constructions; and attributes describing instances of these classes, such as a user's name and year-group, a task's name and description, a construction's content and description, and a session's description and duration.

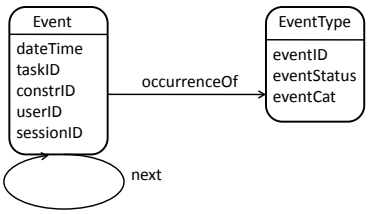

Figure 1: Core Graph Data Model

There is a relationship 'next' linking an instance of Event to the next Event that occurs for the same user, task and session. There is a relationship 'occurrenceOf' linking each instance of Event to an instance of the EventType class.

The instances of the EventType class include: startTask, endTask, numberCreated, numberUnlocked, unlockedNumberChanged, buildingBlockMade, correctModelRuleCreated, incorrectModelRuleCreated, interventionGenerated, interventionShown, in the case of eXpresser (see [8] for the full list); and startTask, endTask, fractionCreated, fractionChanged, fractionReleased, inverventionShown, in the case of FractionsLab.

We see that instances of the EventType class have several attributes, including:

- eventID: a unique numerical identifier for each type of indicator;
- eventStatus: this may be -1, 0 or 1, respectively stating that an occurrence of this type of indicator shows that the student is making negative, neutral or positive progress towards achieving the task goals; an additional status 2 is used for indicators relating to system interventions;
- eventCat: the category into which this indicator type falls; for example, startTask and endTask are task-related indicators; interventionGenerated and interventionShown are system-related ones; numberCreated, numberUnlocked, unlockedNumberChanged are number-related; and fractionCreated, fractionChanged, fractionReleased are fraction-related.

Figure 2 shows a fragment of MiGen interaction data conforming to this graph data model. Specifically, it relates to the interactions of user 5 as he/she is working on task 2 during session 9. The user makes three constructions during this task (with constrIDs 1, 2 and 3). The start and end of the task are delimited by an occurrence of the startTask and endTask indicator type, respectively — events 23041 and 33154. We see that the two events following 23041 relate to an intervention being generated and being shown to the student (this is likely to be because the student was inactive for over a minute after starting the task); following which, the student creates a number — event 24115.

There are additional attributes relating to events, not shown here for simplicity, capturing values relating to the student's

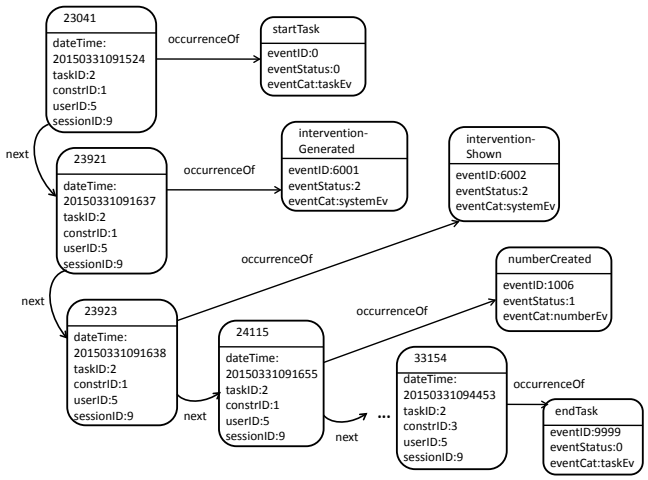

Figure 2: Fragment of Graph Data

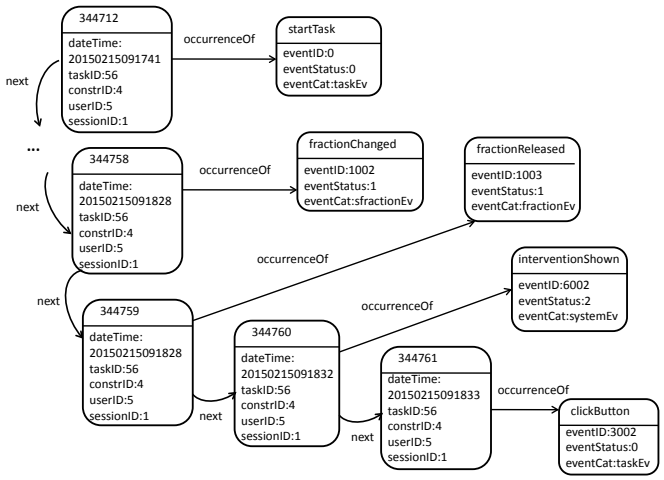

Figure 3: Fragment of Graph Data

constructions and information relating to the system's interventions. For example, for event 24115, the value of the number created, say 5; for event 23921, the feedback strategy used by the system to generate this intervention, say strategy 8; and for event 23923, the content of the message displayed to the user, say "How many green tiles do you need to make your pattern?" and whether this is a high-level interruption by the system or a low-level interruption that the student can choose to view or not. Such information can be captured through additional edges outgoing from an event instance to a literal-valued node: 24115 $\xrightarrow{value}$ 5, 23921 $\xrightarrow{strategy}$ 8, 23932 $\xrightarrow{message}$ "How many green tiles do you need to make your pattern?", 23932 $\xrightarrow{level}$ "high". Since graph data models are semi-structured (and graph data therefore does not need to strictly conform to a single schema), this kind of heterogeneity in the data is readily accommodated.

Figure 3 similarly shows a fragment of FractionsLab interaction data, relating to the interactions of user 5 working on task 56 during session 1. The user makes one construction during this task. We see events relating to the student changing and 'releasing' a fraction. Following which the system displays a message (in this case, it was a high-interruption message of encouragement "Great! Well Done").

We see from Figures 2 and 3 that the sub-graph induced by edges labelled 'next' consists of a set of paths, one path for each task undertaken by a specific user in a specific session. The entire graph is a DAG (directed acyclic graph): there are no cycles induced by the edges labelled 'next' since each links an earlier indicator occurrence to a later one; while the instances of EventType and other literal-valued nodes can have only incoming edges. The entire graph is also a bipartite graph, with the two parts comprising (i) the instances of Event, and (ii) the instances of EventType and the literal-valued nodes.

As a final observation, we note that Figures 1 – 3 adopt a "property graph" notation (e.g. as used in the Neo4J graph database, neo4j.com) in which nodes may have attributes. In a "classical" graph data model, each attribute of a node would be represented by an edge and its value by a literal-valued node. So, for example, the information that the taskID of event 23041 is 2 would be represented by an edge 23041 $\xrightarrow{taskID}$ 2. The query examples in the next section assume this "classical" graph representation.

3. GRAPH QUERIES AND ANALYSES

Because the sub-graph induced by edges labelled 'next' consists of a set of paths, the data readily lends itself to exploration using *conjunctive regular path (CRP) queries* [2]. A CRP query, Q, consisting of n conjuncts is of the form

$$(Z_1, \ldots, Z_m) \leftarrow (X_1, R_1, Y_1), \ldots, (X_n, R_n, Y_n)$$

where each X_i and Y_i is a variable or a constant, each Z_i is a variable that appears also in the right hand side of Q, and each R_i is a regular expression over the set of edge labels. In this context, a regular expression, R, has the following syntax:

$$R := \epsilon \mid a \mid \_ \mid (R_1.R_2) \mid (R_1|R_2) \mid R^* \mid R^+$$

where ϵ denotes the empty string, a denotes an edge label, _ denotes the disjunction of all edge labels, and the operators have their usual meaning. The answer to a CRP query on a graph G is obtained by finding for each $1 \leq i \leq n$ a binary relation r_i over the scheme (X_i, Y_i), where there is a tuple (x, y) in r_i if and only if there is a path from x to y in G such that: $x = X_i$ if X_i is a constant; $y = Y_i$ if Y_i is a constant; and the concatenation of the edge labels in the path satisfies the regular expression R_i. The answer is then given by forming the natural join of the binary relations $r_1, \ldots, r_n$ and finally projecting on $Z_1, \ldots, Z_m$.

To illustrate, the following CRP query returns pairs of events x, y such that x is an intervention message shown to the user by the system and y indicates that the user's next action – in eXpresser – was to create a number (note, variables in queries are distinguished by an initial question mark):

```
(?X,?Y) <- (?X,occurrenceOf,interventionShown),
           (?X,next,?Y),
           (?Y,occurrenceOf,numberCreated)
```

The result would contain pairs such as (23923,24115) from Figure 2, demonstrating that there are indeed situations where an intervention message displayed by the MiGen system leads directly to the creation of a number by the student.

The following query returns pairs of events x, y such that that x is an intervention message shown to the user by the system and y is the user's next action; the type of y is also returned, through the variable `?Z`:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,interventionShown),
              (?X,next,?Y),
              (?Y,occurrenceOf,?Z)
```

The result would contain triples such as (23923,24115,numberCreated) from Figure 2 and (344760,344761,clickButton) from Figure 3, allowing researchers to see what types of events directly follow the display of an intervention message. This would allow the confirmation or contradiction of researchers' expectations regarding the immediate effect of intervention messages on students' behaviours.

Focussing for the rest of this section on the data in Figure 2, the following query returns pairs of events x, y such that x is any type of event and y indicates that the user's next action was to unlock a number; the type of x is also returned, through the variable `?Z`:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,?Z),
              (?X,next,?Y),
              (?Y,occurrenceOf,numberUnlocked)
```

The result would allow researchers to see what types of events immediately precede the unlocking of a number (i.e. the creation of a variable). This would allow confirmation of researchers' expectations about the design of the MiGen system's intelligent support in guiding students towards generalising their models by changing a fixed number to an 'unlocked' one.

The following query returns pairs of events x, y such that that x is an intervention generated by the system and y is any subsequent event linked to x through a path comprising one or more 'next' edges; the type of y is also returned, through the variable `?Z`:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,interventionGenerated),
              (?X,next+,?Y),
              (?Y,occurrenceOf,?Z)
```

The result would contain triples such as (23921, 23923, interventionShown), (23921, 24115, numberCreated), ... (23921, 33154, endTask), allowing researchers to see what types of events directly or indirectly follow the display of an intervention message by the system. This would allow the confirmation or contradiction of researchers' expectations regarding the longer-term effect of intervention messages on students' behaviours.

We can modify the query to retain only pairs x, y that relate to the same construction:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,interventionGenerated),
              (?X,constrID,?C), (?X,next+,?Y),
              (?Y,constrlID,?C), (?Y,occurrenceOf,?Z)
```

The result would contain triples such as
(23921, 23923, interventionShown),
(23921, 24115, numberCreated),
(23921, 24136, numberUnlocked),
(23921, 24189, unlockedNumberChanged),
relating to construction 1 made by user 5 during session 9 for task 2 (two more events — 24136 and 24189 — relating to construction 1 have been assumed here, in addition to 23923 amd 24115 shown in Figure 2, for illustrative purposes). The results would not contain (23921,33154,endTask), since event 33154 relates to construction 3.

To show more clearly the answers to the previous query in the form of possible event *paths*, we can use *extended regular path* (ERP) queries [11], in which a regular expression can be associated with a *path variable* and path variables can appear in the left-hand-side of queries. Thus, for example, the following query returns the possible paths from x to y:

```
(?X,?P,?Y,?Z) <-
      (?X,occurrenceOf,interventionGenerated),
      (?X,constrID,?C), (?X,next+:?P,?Y),
      (?Y,constrID,?C), (?Y,occurrenceOf,?Z)
```

The result would contain answers such as
(23921, [next], 23923, interventionShown),
(23921, [next, 23923, next], 24115, numberCreated),
(23921, [next, 23923, next, 24115, next], 24136, numberUnlocked),
(23921, [next, 23923, next, 24115, next, 24136, next], 24189, unlockedNumberChanged).

The use of the regular expressions `next` and `next+` in the previous queries matches precisely one edge labelled 'next', or any number of such edges (greater than or equal to 1), respectively. However, for finer control and ranking of query answers, it is possible to use *approximate* answering of CRP and ERP queries (see [11, 17]), in which edit operations such as insertion, deletion or substitution of an edge label can be applied to regular expressions.

For example, using the techniques described in [11, 17], the user can chose to allow the insertion of the label 'next' into a regular expression, at an edit cost of 1. Submitting then this query:

```
(?X,?P,?Y,?Z) <-
      (?X,occurrenceOf,interventionGenerated),
      (?X,constrID,?C), APPROX(?X,next:?P,?Y),
      (?Y,constrID,?C), (?Y,occurrenceOf,?Z)
```

would return first exact answers, such as
(23921, [next], 23923, interventionShown). The regular expression `next` in the conjunct `APPROX(?X,next:?P,?Y)` would then be automatically approximated to `next.next`, leading to answers such as

(23921, [next, 23923, next], 24115, numberCreated)
at an edit distance of 1 from the original query. Following this, the regular expression `next.next` would be automatically approximated to `next.next.next`, leading to answers such as
(23921, [next, 23923, next, 24115, next], 24136, numberUnlocked)
at distance 2. This incremental return of paths of increasing length can continue for as long as the user wishes, and allows researchers to examine increasingly longer-term effects of intervention messages on students' behaviours. It would also be possible for users to specify from the outset a minimum and maximum edit distance to be used in approximating and evaluating the query, for example to request paths encompassing between 2 and 4 edges labelled 'next'.

Queries based on evaluating regular expressions over a graph-based representation of interaction data, such as those above, can aid in the exploration of students' behaviours as they are undertaking tasks using ELEs and the effectiveness of the intelligent support being provided by the ELE. The query processing techniques employed are based on incremental query evaluation algorithms which run in polynomial time with respect to the size of the database graph and the size of the query and which return answers in order of increasing edit distance [11]. A recent paper [19] gives details of an implementation, which is based on the construction of an automaton (NFA) for each query conjunct, the incremental construction of a weighted product automaton from each conjunct's automaton and the data graph, and the use of a ranked join to combine answers being incrementally produced from the evaluation of each conjunct. The paper also presents a performance study undertaken on two data sets — lifelong learning data and metadata [17] and YAGO [22]. The first of these has rather 'linear' data, similar to the interaction data discussed here, while the second has 'bushier' connectivity. Query performance is generally better for the former than the latter, and the paper discusses several possible approaches towards query optimisation.

In addition to evaluating queries over the interaction data, by representing the data in the form of a graph it is possible to apply graph structure analyses such as the following:

- *path finding and clustering*: this would be useful for determining patterns of interest across a whole dataset, or focussing on particular students, tasks or sessions c.f. [4];
- *average path length*: this would be useful for determining the amount of student activity (i.e. the number of indicator occurrences being generated per task) across a whole dataset, or focussing on particular students, tasks or sessions;
- *graph diameter*: to determine the greatest distance between any two nodes (which, due to the nature of the data, would be event type nodes); this would be an indication the most long-running and/or most intensive task(s);
- *degree centrality*: determining the in-degree centrality of event type nodes would identify key event types occurring in students' interactions; this analysis could be applied across a whole dataset, or focussing on particular students, tasks or sessions;
- nodes that have a high probability of being visited on a randomly chosen shortest path between two randomly chosen nodes have high *betweenness centrality*; determining this measure for pairs of event type nodes (ignoring the directionality of the 'occurrenceOf' edges) would identify event types that play key mediating roles between other event types.

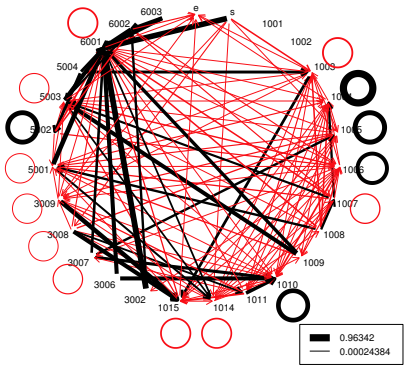

Figure 4: Transitions between Event Types

We have already undertaken some *ad hoc* analyses of interaction data arising from classroom sessions using ELEs. For example, Figure 4 shows the normalised incoming transitions for a 1-hour classroom session involving 22 students using MiGen (in the diagram, s denotes the 'startTask' and e the 'endTask' event types). Event types with an adjacent circle show transitions where this type of event occurs repeatedly in succession. The thickness of each arrow or circle indicates the value of the transition probability: the thicker the line, the higher the probability. Red (light grey) is used for probabilities < 0.2 and black for probabilities ≥ 0.2. We can observe a black arrow $3007 \rightarrow 1005$, indicating transitions from events of type 3007 (detection by the system that the student has made an implausible building block for this task) to events of type 1005 (modification of a rule by the student). Such an observation raises a hypothesis for more detailed analysis or further student observation, namely: "does the construction of an incorrect building block lead students to self-correct their rules?". Developing a better understanding of such complex interaction can lead to improvement of the system. For this particular example, we designed a new prompt that suggests to students to first consider the building block against the given task before proceeding unnecessarily in correcting their rules. More examples of such *ad hoc* analyses are given in [14]. Representing the interaction data in graph form will allow more systematic, flexible and scalable application of graph-structure algorithms such as those identified above.

4. CONCLUSIONS AND FUTURE WORK

We have presented a graph model for representing event-based interaction data arising from Exploratory Learning Environments, drawing on the data generated when students undertake exploratory learning tasks with the eXpresser and

FractionsLab microworlds. Although developed in the context of these systems, the model is a very general one and can easily be used or extended to model similar data from other ELEs.

We have explored the possibilities that evaluating regular path queries over this graph-based representation might provide for exploring the behaviours of students as they are working in the ELE and the effectiveness of the intelligent support that it provides to them. We have also identified additional graph algorithms that may yield further insights about learners, tasks and significant indicators.

Planned worked includes transformation and uploading of the interaction data sets gathered during trials and full classroom sessions of the two systems into an industrial-strength graph database such as Neo4J, following the graph model presented in Section 2; followed by the design, implementation and evaluation of meaningful queries, analyses and visualisations over the graph data, building on the work presented in Section 3. Equipped with an appropriate user interface, educational researchers, designers or even teachers with less technical expertise could in this way explore the data from their perspective. This has the potential to lead to an improved understanding of interaction in this context and to feed back to the design of the ELEs. We see this approach very much in the spirit of "polyglot persistence" (i.e. using different data storage methods to address different data manipulation problems), and hence being used in conjunction with other EDM resources such as DataShop [12]. Another direction of research is investigation of how the flexible querying processing techniques for graph data (including both query approximation and query relaxation) that have been developed in the context of querying lifelong learners' data and metadata [11, 17] might be applied or adapted to the much finer-granularity interaction data described here and the more challenging pedagogical setting of providing effective intelligent support to learners undertaking exploratory tasks in ELEs.

Acknowledgments

This work has been funded by the ESRC/EPSRC MiGen project, the EU FP7 projects iTalk2Learn (#318051) and M C Squared (#610467). We thank all the members of these projects for their help and insights.

5. REFERENCES

[1] N. Belacel, G. Durand, and F. LaPlante. A binary integer programming model for global optimization of learning path discovery. *G-EDM*, 2014.

[2] D. Calvanese and et al. Containment of conjunctive regular path queries with inverse. *KRR*, pages 176–185, 2015.

[3] R. Dekel and K. Gal. On-line plan recognition in exploratory learning environments. *G-EDM*, 2014.

[4] M. Eagle and T. Barnes. Exploring differences in problem solving with data-driven approach maps. *EDM*, 2014.

[5] M. Eagle, D. Hicks, B. Peddycord III, and T. Barnes. Exploring networks of problem-solving interactions. *LAK*, pages 21–30, 2015.

[6] M. Eagle, M. Johnson, and T. Barnes. Interaction networks: Generating high level hints based on network community clustering. *EDM*, 2012.

[7] B. Grawemeyer and et al. Light-bulb moment?: Towards adaptive presentation of feedback based on students' affective state. *IUI*, pages 400–404, 2015.

[8] S. Gutierrez-Santos, E. Geraniou, D. Pearce-Lazard, and A. Poulovassilis. Design of Teacher Assistance Tools in an Exploratory Learning Environment for Algebraic Generalization. *IEEE Trans. Learn. Tech.*, 5(4):366–376, 2012.

[9] S. Gutierrez-Santos, M. Mavrikis, and G. D. Magoulas. A Separation of Concerns for Engineering Intelligent Support for Exploratory Learning Environments. *J. Research and Practice in Inf. Tech.*, 44:347–360, 2013.

[10] A. Harrer, R. Hever, and S. Ziebarth. Empowering researchers to detect interaction patterns in e-collaboration. *Frontiers in Artificial Intelligence and Applications*, 158:503, 2007.

[11] C. Hurtado, A. Poulovassilis, and P. Wood. Finding top-k approximate answers to path queries. *FQAS*, pages 465–476, 2009.

[12] K. Koedinger and et al. A data repository for the EDM community: The PSLC datashop. *Handbook of Educational Data Mining*, 43, 2010.

[13] M. Mavrikis and S. Gutierrez-Santos. Not all Wizards are from Oz: Iterative design of intelligent learning environments by communication capacity tapering. *Computers and Education*, 54(3):641–651, 2010.

[14] M. Mavrikis, Z. Zheng, S. Gutierrez-Santos, and A. Poulovassilis. Visualisation and analysis of students' interaction data in exploratory learning environments. *Workshop on Web-Based Technology for Training and Education (at WWW)*, 2015.

[15] R. Noss and et al. The design of a system to support exploratory learning of algebraic generalisation. *Computers and Education*, 59(1):63–82, 2012.

[16] N. Pinkwart and et al. Graph grammars: An ITS technology for diagram representations. *FLAIRS*, pages 433–438, 2008.

[17] A. Poulovassilis, P. Selmer, and P. Wood. Flexible querying of lifelong learner metadata. *IEEE Trans. Learn. Tech.*, 5(2):117–129, 2012.

[18] O. Scheuer and B. McLaren. CASE: A configurable argumentation support engine. *IEEE Trans. Learn. Tech.*, 6(2):144–157, 2013.

[19] P. Selmer, A. Poulovassilis, and W. P.T. Implementing flexible operators for regular path queries. *GraphQ (EDBT/ICDT Workshops)*, pages 149–156, 2015.

[20] V. Sheshadri, C. Lynch, and T. Barnes. InVis: An EDM tool for graphical rendering and analysis of student interaction data. *G-EDM*, 2014.

[21] J. Stamper, M. Eagle, T. Barnes, and M. Croy. Experimental evaluation of automatic hint generation for a logic tutor. *Artificial Intelligence in Education*, pages 345–352, 2011.

[22] F. Suchanek, G. Kasneci, and G. Weikum. YAGO: a core of semantic knowledge. *WWW*, 2007.

[23] D. Suthers. From contingencies to network-level phenomena: Multilevel analysis of activity and actors in heterogeneous networked learning environments. *LAK*, 2015.